Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 30 (2012) 274 – 282

Procedia Engineering

www.elsevier.com/locate/procedia

International Conference Communication Technology and System Design 2011

Design of Low Bandwidth Peripherals Using High Performance Bus Architecture

Roopa.M[a],Vani.R.M[b],P.V.Hunagund[c], a*

[a]*Department of Electronics & Communication,DayanandaSagar College of Engineering,Bangalore,India*
[b]*University Science Instrumentation Center,Gulbarga University,Gulbarga,India*
[c]*Department of Applied Electronics, Gulbarga University,Gulbarga,India*

Abstract

System on-chip (SOC) communication has a significant impact on system performance, power dissipation and time to market. System designers, as well as the researchers community focuses on low power dissipation. A reliable on-chip communication standard is a must in any SOC. The AMBA 2.0 APB is a peripheral bus standard for low bandwidth peripheral. In this paper, we present the design of APB controller which handles the transactions between the master and peripheral devices. The final design which integrates the peripheral devices with the APB controller and simulated.

Keywords: Advanced microcontroller bus architecture;Advanced Peripheral Bus;System On Chip.

1. Introduction

The increasing complexity of Systems-on-Chip (SOC) has led to the critical "design productivity gap" problem. On-chip communication architectures have a significant impact on system performance, power dissipation and time-to-market. System designers, as well as the research community have focused on the issue of exploring, evaluating, and designing SOC communication architectures to meet the targeted design goals [1].

2. Advanced Peripheral Bus

The APB is part of the AMBA hierarchy of buses and is optimized for minimal power consumption and reduced interface complexity. The AMBA APB is used to interface to any peripherals which are low

* Roopa.M. Tel.: +91-9449229242.
E-mail address: surajroopa@gmail.com.

doi:10.1016/j.proeng.2012.01.861

bandwidth and do not require the high performance of a pipelined bus interface [2].

2.1 APB Specifications:

The APB specification is described under three sections as shown in headings State diagram, Write transfer and Read transfer.

- State Diagram

The state diagram, shown in Figure 2.1, can be

Used to represent the activity of the peripheral bus. Operation of the state machine is through the three States described below,

Idle: The default state for the peripheral bus.

Setup: When a transfer is required the bus moves into the SETUP state, where the appropriate select signal, **PSELx** is asserted. The bus only remains in the SETUP state for one clock cycle and will always move to the ENABLE state on the next rising edge of the clock.

Enable: In the ENABLE state the enable signal, **PENABLE** is asserted. The address, write and select signals all remain stable during the transition from the SETUP to ENABLE state. The ENABLE state also lasts for a single clock cycle and after this state the bus will return to the IDLE state if no further transfers are required. Alternatively, if another transfer is to follow then the bus will move directly to the SETUP state. It is acceptable for the address, write and select signals to glitch during a transition from the ENABLE to SETUP states.

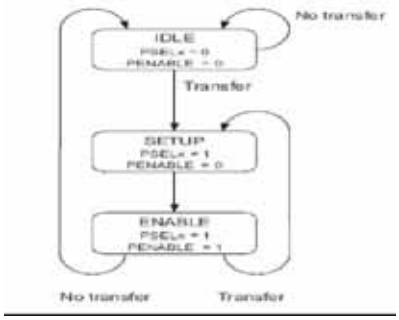

Fig: 2.1: APB Controller State Diagram,

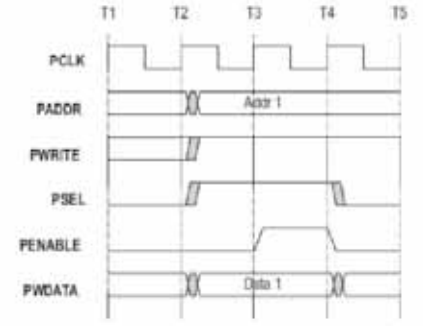

Fig 2.2: Write Cycle of APB Controller,

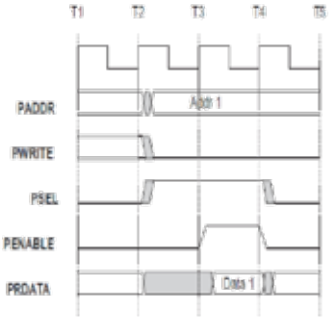

Fig: 2.3: Read cycle of APB Controller

- Write Transfer

The basic write transfer of an APB is shown in Figure 2.2.The write transfer starts with the address, write data, write signal and select signal all changing after the rising edge of the clock. The first clock cycle of the transfer is called the SETUP cycle. After the following clock edge the enable signal **PENABLE** is asserted and this indicates that the ENABLE cycle is taking place. The address, data and control signals all remain valid throughout the ENABLE cycle. The transfer completes at the end of this cycle [3].

- Read Transfer

The Figure 2.3 shows a read transfer depicting the clock and control signal.The timing of the address, write, select and strobe signals are all the same as for the write transfer. In this case of a read, the slave must provide the data during the ENABLE cycle. The data is sampled on the rising edge of clock at the end of the ENABLE cycle.

3. Architecture Overview

The architecture is developed based on the standard specifications of the protocol [2]. The architecture explains the function modules designed.

3.1 Design Module

The design module is a result of integration of low bandwidth peripheral devices with the APB controller. The operation of the slave devices are initiated and controlled by APB controller. The Top module is presented first and the sub modules are discussed in further sections.

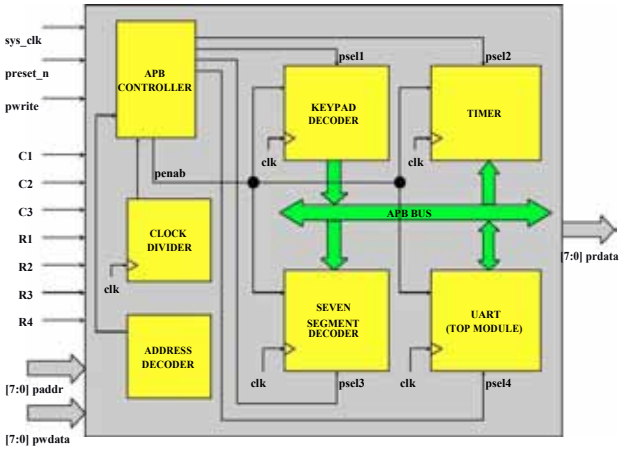

Figure 3.1: APB Controller Design Block Diagram

The block diagram is depicted in figure 3.1 have major control signals between the APB controller and the other modules as indicated. APB Controller is unit responsible for the flow of data and control signals between the Slaves and the Master. The different peripherals modules are UART, Timer, Seven Segment Decoder and Keypad Decoder. Input for the APB controller is available from the AHB-to-APB Bridge which acts as the master for the APB. The APB switches data between the slaves and the master using the control signals.

The Hierarchy of designs is as shown below:

➔ **TOP MODULE** – APB controller integrated with peripherals (Slaves)
- APB Controller.
- UART Top Module
  - UART Transmitter
  - UART Receiver
  - Baud Rate Generator
- Seven Segment Decoder
- Timer
- Keypad Decoder

3.2 Top Module – APB Integrated with Peripheral Devices.

Table 3.1: Signal list of top module design block.

I/O NAMES	I/O DESCRIPTION	I/O DESCRIPTION
sys_clk	input	System clock.
preset_n	input	System reset signal.
pwrite	input	Read or Write signal.
C1	input	Column 1 input.
C2	input	Column 2 input.
C3	input	Column 3 input.
R1	input	Row 1 input.
R2	input	Row 2 input.
R3	input	Row 3 input.
R4	input	Row 4 input.
paddr [7:0]	input	8 bit address bus.
pwdata [7:0]	input	8 bit Write Data Bus.
prdata [7:0]	output	8 bit Read Data Bus.

Figure 3.1 depicts the block diagram of APB Top Module. The signals responsible for functioning of the module are listed in table 3.1. Top Module integrates all the peripheral devices with the APB Controller. The"sys_clk" is the system clock signal and the "preset_n" is the system reset signal for the unit. The "paddr" is 8 bit address available from the master AHB-to-APB Bridge. The "pwdata" signal is the 8 bit write data bus. The "pwrite" is the Read/Write signal. If pwrite = 1, it's a write operation else read operation. C1, C2, C3 and R1, R2, R3, R4 are Column and Row inputs from the 4X3 keypad matrix. The "prdata" is the 8 bit read bus. When an address is placed, the APB controller selects the slave device which needs a transaction. The operation is enabled by the controller and read or writes transaction takes place.

3.3 APB Controller

The APB controller acts as the intermediate unit between master AHB-to-APB Bridge, and the APB slaves. The slave devices are selected using the select lines from the APB controller and its operation is enabled using the enable signal. The block diagram of the APB controller is as shown in figure 3.2. The APB Controller works according to state diagram as in the figure 2.1.

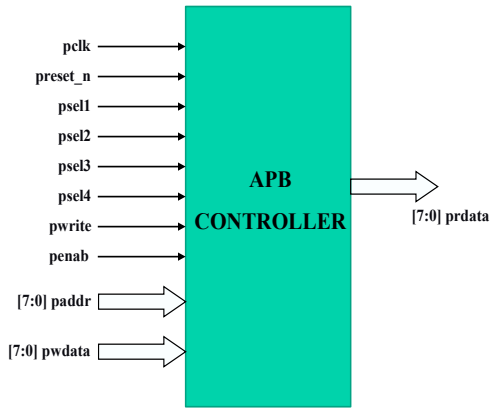

Figure 3.2: Block Diagram of APB Controller.

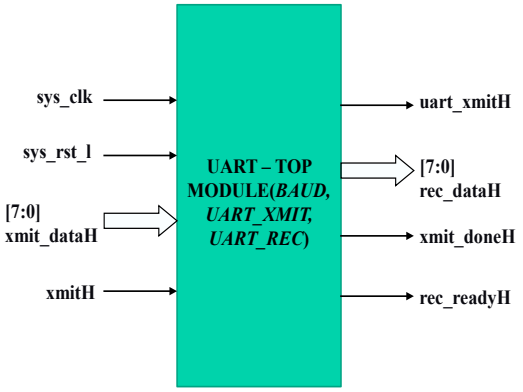

Figure 3.3: Block Diagram of UART Top Module.

The Controller works with clock "pclk" and has "preset_n" as the system reset signal. The data read slave places its data on the "prdata" bus and data is written to slave device through "pwdata" bus. The state machine has 3 states IDLE, SETUP and ENABLE. Initially the controller stays in the IDLE state. If any transfer is then the controller moves to the SETUP state. The SETUP state lasts for only one clock cycle and the controller moves to the ENABLE state. The operation of the peripheral device is enabled in the ENABLE state by issuing the enable signal in next signal [4].

3.3 UART Top Module

The UART is a serial transmission and reception unit where the transmitter and receiver works asynchronously with different clock signals. The block diagram of the UART top module is depicted in the figure 3.3.The Top module UART design includes three sub-modules i.e. UART Transmitter, UART Receiver and Baud Rate Generator. The top module takes "sys_clk" as input and the Transmitter and Receiver works with the "uart_clk" which is generated by the baud rate generator. The UART Transmitter transmits the loaded data (8 Bit) serially through "uart_xmitH" line when the "xmitH" signal is asserted. After the transmission is complete, "xmit_doneH" signal is issued.

The UART Top module is configured in such a way that the transmitter line of UART Transmitter is connected to the receiver line of the UART Receiver module. Input data at the Receiver is synchronized and sampled by the internal units. The serial data is converted to parallel and stored in a register"rec_dataH". After the reception operation is complete the "rec_readyH" signal goes high. The individual blocks of the transmitter and receiver are in further sections.

- UART Transmitter

The UART transmitter as an individual unit starts its operation when xmitH signal goes high. The transmitter module has internal units such as Parallel to Serial converter, counters, start data and stop bit, mux and a control unit (state machine). The Parallel to serial converter or Serialize unit, a shift the loaded 8 bit data one bit at a time and transmits for a particular baud rate. The counters keep track of the number of bits of data that are transmitted. The state machine controls the data flow and the operation among the different units in the transmitter. The state machine issues the control signals among the units to achieve a proper transmission.

- Baud Rate Generator

The receiver and transmitter works with the UART clock which is provided by this unit. The baud rate is fed to this unit for which the receiver and transmitter works. The baud rate generator internally consists of a counter which is loaded with the baud rate data. The UART clock is the output of the unit which switches its value for every reset of the counter. This results in a clock signal which is fed to transmitter and the receiver.

- UART Receiver

It has a dual rank synchronizer, bit cell counter, received bit counter, de-serializer (serial to parallel converter) and a state machine as its units. The state machine is the control unit which controls all the operations in the receiver unit by issuing the control signals. The receiver takes UART clock as the reference clock signal. The serial incoming data for this unit is converted in to parallel and stored in the parallel register.

3.5 Seven Segment Decoder

The Seven Segment Decoder is a simple decoder Module which takes ASCII code for corresponding number and characters (# and *) in hexadecimal form and gives the seven segment code for corresponding number and character.

3.6 Timer

The timer module presented is a countdown timer which gives a time out signal after a particular amount of time. The time duration at which the time out signal is to be issued is loaded in the form of input data to the timer. This data is loaded onto the counter register and the countdown operation starts on the control signal.

The Timer unit is an 8 bit countdown timer. The Timer module consists of a data path and a control unit. The data path unit is the one where the data are processed and the control unit controls the data process by using the control signals. The data path unit contains of two counters: “fast_counter” and “counter”. The “counter” determines the time out interval. The initial data (8 bit) that has to be loaded to the timer is fed through “SW” input. On “start” signal the countdown operation starts. The fast_counter is a continuous counter. The loaded data will be moved to the “counter” register. For one complete count of “fast_counter” i.e. 256 counts, the “counter” will be decremented by one. This count continues till the “counter” register will become zero. After the “counter” becomes zero a “time_out” signal is issues which is the output of the timer unit.

3.7 Keypad Decoder

The Keypad Decoder is an input device which reads values or data from the user. The user or programmer can enter the input through a keypad matrix (4X3 matrix is used in the present design). The Keypad Decoder unit takes combination the Rows and Columns i.e. R1, R2, R3, R4, C1, C2, and C3 as input from the keypad 4X3 matrix and gives the ASCII code (8 -bit) for corresponding number or character (# and *) in hexadecimal form.

4. Design Module

4.1 Top Module:

The figure 4.1 below shows the simulation result of The Top Module which integrates all the design with the APB controller.

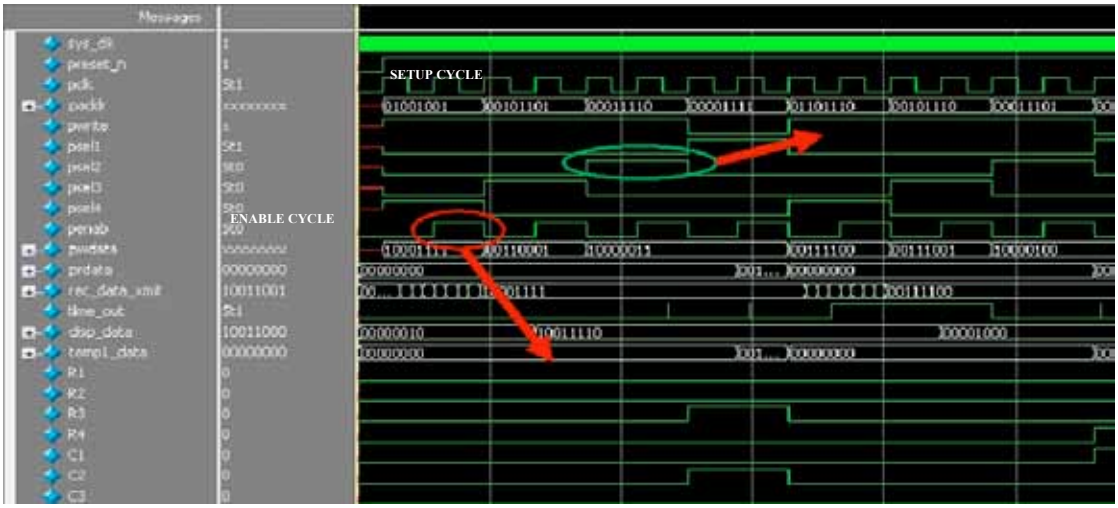

Figure 4.1: Simulation result of Top Module.

4.2 UART Top Module:

The simulation result of the UART top module is Shown in the figure 4.2. The data received is same as the data transmitted as the transmitter line is connected to the receiver line for implementation purpose.

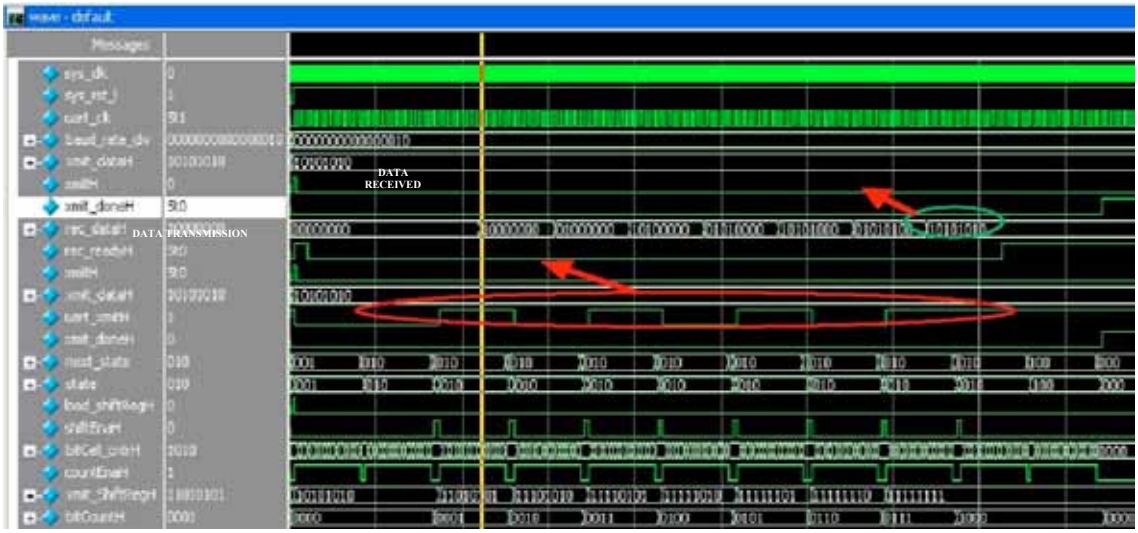

Figure 4.2: Simulation result of the UART top Module.

4.3 *Seven Segment Decoder Module:*

The figure 4.3 shows the simulation result of seven Segment Decoder module.

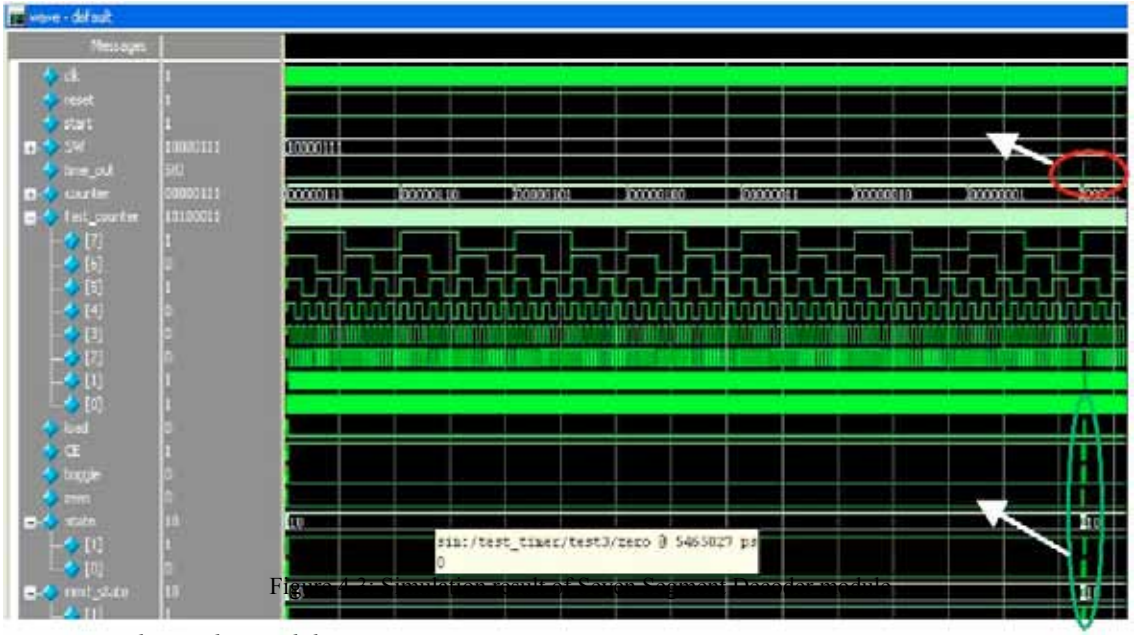

Figure 4.3: Simulation result of Seven Segment Decoder module

4.4 Keypad Decoder Module

The figure 4.4 shows the simulation relation of the Keypad Decoder module. The acknowledge signal and the send signal are indicated in the waveform

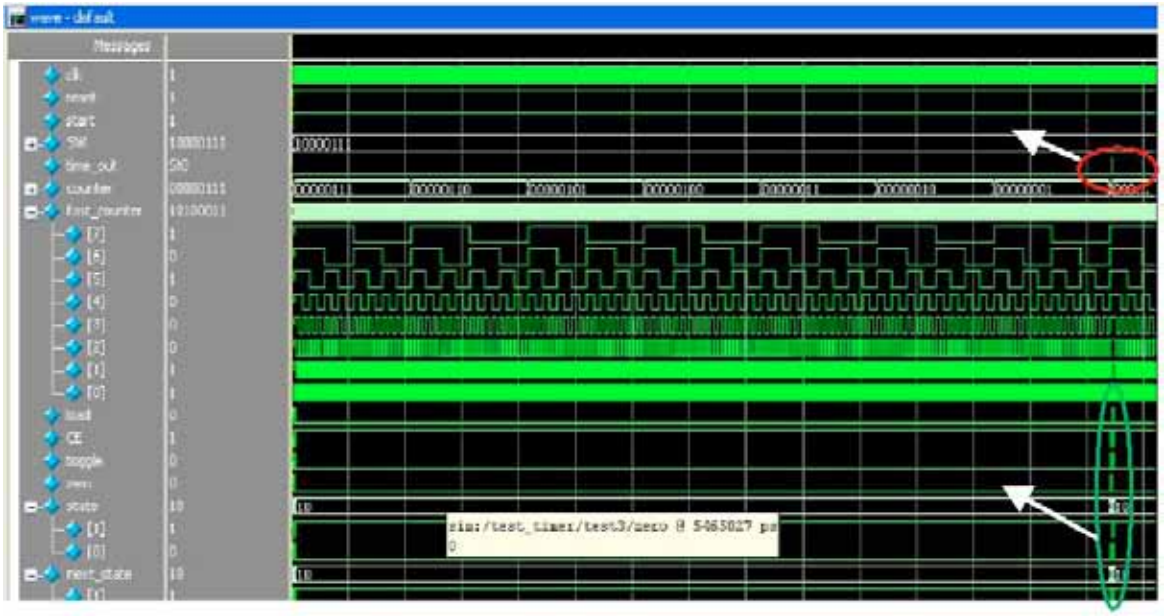

Figure 4.4: Simulation result of Keypad Decoder

5. Conclusion

AMBA APB Protocol has been designed with low Bandwidth Peripherals. The Read and Write Operation are performed on peripherals and controlled by APB Controller. The read or write transaction is completed in two bus cycles when a particular slave is selected by select line and the operation is enabled at the latch of enable signal.

References

[1] Nikil Dutt, Kaustav Banerjee, Luca Benini, Kanishka Lahiri, Sudeep Pasricha, "Tutorial 5: SoC Communication Architectures: Technology, Current Practice, Research, and Trends", *vlsid, pp.8, 20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07),* 2007.

[2] AMBA *Specification* (Rev 2.0).

[3] Flynn, D. Adv. RISC Machines Ltd., Cambridge, "AMBA: enabling reusable on-chip designs*", IEEE Micro,* Publication Date: Jul/Aug1997.

[4] *"*Using formal techniques to Debug the AMBA System-on-Chip Bus Protocol*", Proceedings of the Design, Automation and Test in Europe Conference and Exhibition,* 2003.

[5] Xilinx ISE *Synthesis and Verification Design Guide.*

[6] Sameer Palnitkar, Verilog HDL, *A Guide to Digital Design and Synthesis, 2nd Edition, 2008.*

Morris Mano, Digital Logic and Computer Design, PHI, 2005.